Deviation Estimation between Distributed Data Streams
Emmanuelle Anceaume, Yann Busnel

To cite this version:
Emmanuelle Anceaume, Yann Busnel. Deviation Estimation between Distributed Data Streams. 10th European Dependable Computing Conference (EDCC 2014), May 2014, Newcastle, United Kingdom. pp.35-45, 10.1109/EDCC.2014.27. hal-00998702

HAL Id: hal-00998702
https://hal.archives-ouvertes.fr/hal-00998702
Submitted on 2 Jun 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Deviations Estimation between Distributed Data Streams

Emmanuelle Anceaume  
IRISA / CNRS  
Rennes, France  
Emmanuelle.Anceaume@irisa.fr

Yann Busnel  
LINA / Université de Nantes  
Nantes, France  
Yann.Busnel@univ-nantes.fr

Abstract—The analysis of massive data streams is fundamental in many monitoring applications. In particular, for network operators, it is a recurrent and crucial issue to determine whether huge data streams, received at their monitored devices, are correlated or not as it may reveal the presence of malicious activities in the network system. We propose a metric, called codeviation, that allows to evaluate the correlation between distributed streams. This metric is inspired from classical metric in statistics and probability theory, and as such allows us to understand how observed quantities change together, and in which proportion. We then propose to estimate the codeviation in the data stream model. In this model, functions are estimated on a huge sequence of data items, in an online fashion, and with a very small amount of memory with respect to both the size of the input stream and the values domain from which data items are drawn. We give upper and lower bounds on the quality of the codeviation, and provide both local and distributed algorithms that additively approximates the codeviation among \( n \) data streams by using \( O((1/\varepsilon) \log(1/\delta) (\log N + \log m)) \) bits of space for each of the \( n \) nodes, where \( N \) is the domain value from which data items are drawn, and \( m \) is the maximal stream’s length. To the best of our knowledge, such a metric has never been proposed so far.

Index Terms—Data stream model; correlation metric; distributed approximation algorithm; DDoS attacks.

I. INTRODUCTION AND BACKGROUND

Performance of many complex monitoring applications, including Internet monitoring applications, data mining, sensors networks, network intrusion/anomalies detection applications, depend on the detection of correlated events. For instance, detecting correlated network anomalies should drastically reduce the number of false positive or negative alerts that networks operators have to currently face when using network management tools such as SNMP or NetFlow. Indeed, to cope with the complexity and the amount of raw data, current network management tools analyze their input streams in isolation [1], [2]. Diagnosing flooding attacks through the detection of correlated flows should improve intrusions detection tools as proposed in [3], [4], [5]. In the same way, analyzing the effect of multivariate correlation for an early detection of Distributed Denial of Service (DDoS) is shown in [6]. The point is that, in all these monitoring applications, data streams arrive at nodes in a very high rate and may contain up to several billions of data items per day. Thus computing statistics with traditional methods is unpractical due to constraints on both available processing capacity, and memory. The problem tackled in this paper is the on-line estimation of data streams correlation. More precisely, we propose a distributed algorithm that approximates with guaranteed error bounds in a single pass the linear relation between massive distributed sequences of data.

Two main approaches exist to monitor in real time massive data streams. The first one consists in regularly sampling the input streams so that only a limited amount of data items is locally kept. This allows to exactly compute functions on these samples. However, accuracy of this computation with respect to the stream in its entirety fully depends on the volume of data items that has been sampled and their order in the stream. Furthermore, an adversary may easily take advantage of the sampling policy to hide its attacks among data items that are not sampled, or in a way that prevents its “malicious” data items from being correlated [7]. In contrast, the streaming approach consists in scanning each piece of data of the input stream on the fly, and in locally keeping only compact synopses or sketches that contain the most important information about these data. This approach enables to derive some data streams statistics with guaranteed error bounds without making any assumptions on the order in which data items are received at nodes. Most of the research done so far with this approach has focused on computing functions or statistics measures with error \( \varepsilon \) using \( \text{poly}(1/\varepsilon, \log n) \) bits of space where \( n \) is the domain size of the data items. These include the computation of the number of different data items in a given stream [8], [9], [10], the frequency moments [11], the most frequent data items [11], [12], the entropy of the stream [13], [14], [15], or the information divergence over streams [16].

On the other hand, very few works have tackled the distributed streaming model, also called the functional monitoring problem [17], which combines features of both the streaming model and communication complexity models. As in the streaming model, the input data is read on the fly, and processed with a minimum workspace and time. In the communication complexity model, each node receives an input data stream, performs some local computation, and communicates only with a coordinator who wishes to continuously compute or estimate a given function of the union of all the input streams. The challenging issue in this model is for the coordinator to compute the given function by minimizing the

\[ \text{Deviation Estimation between Distributed Data Streams} \]
number of communicated bits [17], [18], [19]. Cormode et al. [17] pioneered the formal study of functions in this model by focusing on the estimation of the first three frequency moments \( F_0, F_1 \) and \( F_2 \) [11]. Arackaparambil et al. [18] considered the empirical entropy estimation [11] and improve the work of Cormode by providing lower bounds on the frequency moments, and finally distributed algorithms for counting at any time \( t \) the number of items that have been received by a set of nodes from the inception of their streams have been proposed in [20], [21].

In this paper, we go a step further by studying the dispersion matrix of distributed streams. Specifically, we propose a novel metric that allows us to approximate in real time the correlation between distributed and massive streams. This metric, called the sketch codeviation, allows us to quantify how observed data items change together, and in which proportion. As shown in [22], such a network-wide traffic monitoring tool should exist a specific node, called the \textit{coordinator} in the following, with which each node may communicate [17]. We assume that communication is instantaneous. We refer the reader to [23] for a detailed description of data streaming models and algorithms.

**B. Preliminaries**

We first present notations and background that make this paper self-contained. Let \( \sigma \) be a stream of data items that arrive sequentially. Each data item \( i \) is drawn from the universe \( \Omega = \{1, 2, \ldots, N\} \), where \( N \) is very large. A natural approach to study a data stream \( \sigma \) of length \( m' \) is to model it as a fingerprint vector (or item frequency vector) over the universe \( \Omega \), given by \( X = (x_1, x_2, \ldots, x_N) \) where \( x_i \) represents the number of occurrences of data item \( i \) in \( \sigma \). Note that \( 0 \leq x_i \leq m' \). We have \( \|X\|_1 = \sum_{i\in\Omega} x_i \), i.e., \( \|X\|_1 \) is the norm of \( X \). Thus \( m' = \|X\|_1 \).

1) \textit{Codeviation:} In this paper, we focus on the computation of the deviation between any two streams using a space efficient algorithm with some error guarantee. The extension to a distributed environment \( \sigma_1, \ldots, \sigma_n \) is studied in Section V. We propose a metric over fingerprint vectors of items, which is inspired from the classical covariance metric in statistics. Such a metric allows us to qualify the dependence or correlation between two quantities by comparing their variations. As will be shown in Section VI, this metric captures shifts in the network-wide traffic behavior when a DDoS attack is active. The codeviation between any two fingerprint vectors \( X = (x_1, x_2, \ldots, x_N) \), and \( Y = (y_1, y_2, \ldots, y_N) \) is the real number denoted \( \text{cod}(X, Y) \) defined by

\[
\text{cod}(X, Y) = \frac{1}{N} \sum_{i\in\Omega} (x_i - \overline{x})(y_i - \overline{y}) = \frac{1}{N} \sum_{i\in\Omega} x_i y_i - \overline{x} \overline{y} \quad (1)
\]

where \( \overline{x} = \frac{1}{N} \sum_{i\in\Omega} x_i \) and \( \overline{y} = \frac{1}{N} \sum_{i\in\Omega} y_i \).

2) \textit{2-universal Hash Functions:} In the following, we use hash functions randomly picked from a 2-universal hash family. A collection \( H \) of hash functions \( h : \{1, \ldots, M\} \rightarrow \{0, \ldots, M'\} \) is said to be 2-universal if for every \( h \in H \) and for every two different items \( x, y \in [M] \), \( \mathbb{P}\{h(x) = h(y)\} \leq \frac{1}{M'} \), which is exactly the probability of collision obtained if the hash function assigns truly random values to any \( x \in [M] \).

3) \textit{Randomized \((\epsilon, \delta)\)-additively-approximation Algorithm:} A randomized algorithm \( A \) is said to be an \((\epsilon, \delta)\)-additively-approximation of a function \( \phi \) on \( \sigma \) if, for any sequence of items in the input stream \( \sigma \), \( A \) outputs \( \hat{\phi} \) such that \( \mathbb{P}\{|\hat{\phi} - \phi| > \epsilon\} < \delta \), where \( \epsilon, \delta > 0 \) are given as parameters of the algorithm.
III. Sketch codeviation

As presented in the Introduction, we propose a statistic tool, named the sketch codeviation, which allows to approximate the codeviation between any two data streams using compact synopses or sketches. We then give bounds on the quality of this tool with respect to the computation of the codeviation applied on full streams.

**Definition 1** (Sketch codeviation) Let $X$ and $Y$ be any two fingerprint vectors of items, such that $X = (x_1, \ldots, x_N)$ and $Y = (y_1, \ldots, y_N)$. Given a precision parameter $k$, we define the sketch codeviation between $X$ and $Y$ as

$$
\widehat{\text{cod}}_k(X,Y) = \min_{\rho \in \mathcal{P}_k(\Omega)} \text{cod}(\hat{X}_\rho, \hat{Y}_\rho)
$$

where $\forall a \in \rho$, $\hat{X}_\rho(a) = \sum_{i \in a} x_i$, and $\mathcal{P}_k(\Omega)$ is a $k$-cell partition of $\Omega$, i.e., the set of all the partitions of the set $\Omega$ into exactly $k$ nonempty and mutually disjoint sets (or cells).

**Lemma 2** Let $X = (x_1, \ldots, x_N)$, and $Y = (y_1, \ldots, y_N)$ be any two fingerprint vectors. We have

$$
\widehat{\text{cod}}_N(X,Y) = \text{cod}(X,Y)
$$

**Proof:** It exists a unique partition $\rho_N$ of $N$ into exactly $N$ nonempty and mutually disjoint sets, such that $\rho_N$ is made of $N$ singletons $\rho_N = \{\{1\}, \{2\}, \ldots, \{N\}\}$. Thus for any cell $a \in \rho_N$, there exists a unique $i \in \Omega$ such that $\hat{X}_\rho(a) = x_i$. Thus, $\hat{X}_\rho = X$ and $\hat{Y}_\rho = Y$.

Note that for $k > N$, it does not exist a partition of $N$ into $k$ nonempty parts. By convention, for $k > N$, $\widehat{\text{cod}}_k(X,Y) = \widehat{\text{cod}}_N(X,Y)$.

**Proposition 3** The sketch codeviation is a function of the codeviation. We have

$$
\widehat{\text{cod}}_k(X,Y) = \text{cod}(X,Y) + \mathcal{E}_k(X,Y)
$$

where $\mathcal{E}_k(X,Y) = \min_{\rho \in \mathcal{P}_k(\Omega)} \frac{1}{N} \sum_{a \in \rho} \sum_{i \in a} \sum_{j \in a \setminus \{i\}} x_i y_j$.

**Proof:** From Relation (1), we have

$$
\widehat{\text{cod}}_k(X,Y) = \min_{\rho \in \mathcal{P}_k(\Omega)} \left( \frac{1}{N} \sum_{a \in \rho} \hat{X}_\rho(a) \hat{Y}_\rho(a) \right)
$$

which concludes the proof.

The value $\mathcal{E}_k(X,Y)$ (which corresponds to the minimum sums over any partition $\rho$ in $\mathcal{P}_k(\Omega)$) represents the overestimation factor of the sketch codeviation with respect to the codeviation.

A. Derivation of Lower Bounds on $\mathcal{E}_k(X,Y)$

We first show that if $k$ is large enough, then the overestimation factor $\mathcal{E}_k(X,Y)$ is null, that is, the sketch codeviation matches exactly the codeviation.

**Theorem 4** (Accuracy of the sketch codeviation) Let $X$ and $Y$ be any two fingerprint vectors of items, such that $X = (x_1, \ldots, x_N)$ and $Y = (y_1, \ldots, y_N)$. Then $\widehat{\text{cod}}_k(X,Y) = \text{cod}(X,Y)$ if

$$
k \geq \left| \text{supp}(X) \cap \text{supp}(Y) \right| + \mathbf{1}_{\text{supp}(X) \setminus \text{supp}(Y)} + \mathbf{1}_{\text{supp}(Y) \setminus \text{supp}(X)}
$$

where $\text{supp}(X)$, respectively $\text{supp}(Y)$, represents the support of distribution $X$, respectively $Y$ (i.e., the set of items in $\Omega$ that have a non null frequency $x_i \neq 0$, respectively $y_i \neq 0$, for $1 \leq i \leq N$), and notation $\mathbf{1}_A$ denotes the indicator function which is equal to 1 if the set $A$ is not empty and 0 otherwise.

**Proof:** Two cases are examined.

**Case 1:**

Let $k = \left| \text{supp}(X) \cap \text{supp}(Y) \right| + \mathbf{1}_{\text{supp}(X) \setminus \text{supp}(Y)} + \mathbf{1}_{\text{supp}(Y) \setminus \text{supp}(X)}$. We consider a partition $\overline{\rho} \in \mathcal{P}_k(\Omega)$ defined as follows

$$
\begin{align*}
\forall \ell & \in \text{supp}(X) \cap \text{supp}(Y), \{\ell\} \in \overline{\rho} \\
\text{supp}(X) \setminus \text{supp}(Y) & \in \overline{\rho} \\
\text{supp}(Y) \setminus \text{supp}(X) & \in \overline{\rho}
\end{align*}
$$

(2)
Then from Relation (2) we have
\[
\begin{align*}
\forall \ell \in \text{supp}(X) \cap \text{supp}(Y), & \quad \sum_{i \notin \{\ell\}} \sum_{j \notin \{\ell\}} x_i y_j = 0 \\
\forall \ell \in \text{supp}(X) \setminus \text{supp}(Y), & \quad y_\ell = 0 \\
\forall \ell \in \text{supp}(X)^c, & \quad x_\ell = 0.
\end{align*}
\]
Thus, \( \sum_{a \in \Omega} \sum_{i \in a} \sum_{j \notin a \setminus \{i\}} x_i y_j = 0 \). From Proposition (3), we get that \( \text{cod}_k(X,Y) = \text{cod}(X,Y) \).

**Case 2:**
For \( k > 1 \) \( |\text{supp}(X) \cap \text{supp}(Y)| > |\text{supp}(X) \setminus \text{supp}(Y) + 1| \text{supp}(Y) \setminus \text{supp}(X) \) (and \( k < N \)), it is always possible to split one of the two last cells of \( \Omega \) as defined in Relation (2) with a singleton \( \{\ell\} \) such that \( x_\ell = 0 \) or \( y_\ell = 0 \).

Both cases complete the proof.

**B. Derivation of Upper Bounds on \( \mathcal{E}_k(X,Y) \)**

We have shown with Theorem 4 that the sketch codeviation matches exactly the codeviation if \( k \geq |\text{supp}(X) \cap \text{supp}(Y)| + 1 |\text{supp}(X) \setminus \text{supp}(Y) + 1| \text{supp}(Y) \setminus \text{supp}(X) \). In this section, we characterize the upper bound of the overestimation factor, i.e., the error made with respect to the codeviation, when \( k \) is strictly less than this bound. To prevent problems of measurability, we restrict the classes of fingerprint vector under consideration. Specifically, given \( m_X \) and \( m_Y \) any positive integers, we define the classes \( \mathcal{X} \) and \( \mathcal{Y} \) as \( \mathcal{X} = \{X = (x_1, \ldots, x_N)\} \) such that \( || X ||_1 = m_X \) and \( \mathcal{Y} = \{Y = (y_1, \ldots, y_N)\} \) such that \( || Y ||_1 = m_Y \). The following theorem derives the maximum value of the overestimation factor.

**Theorem 5 (Upper bound of \( \mathcal{E}_k(X,Y) \))** Let \( k \geq 1 \) be the precision parameter of the sketch codeviation. For any two fingerprint vectors \( X \in \mathcal{X} \) and \( Y \in \mathcal{Y} \), let \( \mathcal{E}_k \) be the maximum value of the overestimation factor \( \mathcal{E}_k(X,Y) \). Then, the following relation holds.

\[
\mathcal{E}_k = \max_{X \in \mathcal{X}, Y \in \mathcal{Y}} \mathcal{E}_k(X,Y) = \begin{cases} m_X m_Y \frac{N}{k} & \text{if } k = 1, \\ m_X m_Y \left( 1 - \frac{1}{N} \right) & \text{if } k > 1. \end{cases}
\]

**Proof:** The first part of the proof is directly derived from Lemma 6. Using Lemmata 7 and 8, we obtain the statement of the theorem.

**Lemma 6** For any two fingerprint vectors \( X \in \mathcal{X} \) and \( Y \in \mathcal{Y} \), the maximum value \( \mathcal{E}_1 \) of the overestimation factor is exactly

\[
\mathcal{E}_1 = \max_{X \in \mathcal{X}, Y \in \mathcal{Y}} \mathcal{E}_1(X,Y) = m_X m_Y \frac{N}{N}. \]

**Proof:** \( \forall X \in \mathcal{X}, \forall Y \in \mathcal{Y} \), we are looking for the maximal value of \( \mathcal{E}_1(X,Y) \) under the following constraints:

\[
\begin{align*}
0 \leq x_i \leq m_X & \quad \text{with } 1 \leq i \leq N, \\
0 \leq y_i \leq m_Y & \quad \text{with } 1 \leq i \leq N, \\
\sum_{i=1}^{N} x_i &= m_X, \\
\sum_{i=1}^{N} y_i &= m_Y.
\end{align*}
\]

In order to relax one constraint, we set \( x_N = m_X - \sum_{i=1}^{N-1} x_i \). We rewrite \( \mathcal{E}_1(X,Y) \) as a function \( f \) such that

\[
f(x_1, \ldots, x_{N-1}, y_1, \ldots, y_N) = \sum_{i=1}^{N-1} \sum_{j=1, j \neq i}^{N} y_j + \left( m_X - \sum_{i=1}^{N-1} x_i \right) \sum_{i=1}^{N-1} y_i.
\]

The function \( f \) is differentiable on its domain \( [0..m_X]^{N-1} \times [0..m_Y]^N \). Thus we get

\[
df(x_1, \ldots, x_{N-1}, y_1, \ldots, y_N) = \sum_{i=1}^{N-1} \sum_{j=1, j \neq i}^{N} y_j - \sum_{j=1}^{N} y_j = y_N - y_i.
\]

We need to consider the following two cases:

1. \( y_N > y_i \). Function \( f \) is strictly increasing, and its maximum is reached for \( x_1 = m_X \) (\( f \) is a Schur-convex function). By Relation 3, \( \forall j \in \mathcal{Y} \setminus \{i\}, x_j = 0 \).
2. \( y_N \leq y_i \). Function \( f \) is decreasing, and its minimum is reached at \( x_1 = 0 \).

By symmetry on \( Y \), the maximum of \( \mathcal{E}_1(X,Y) \) is reached for a distribution in which exactly one \( y_i \) is equal to \( m_Y \), and all the others \( y_j \) are equal to zero, which corresponds to the Dirac distribution. On the other hand, if the spike element of \( Y \) is the same as the one of \( X \), then \( \mathcal{E}_1(X,Y) = 0 \), which is clearly not the maximum.

Thus, for all \( X \in \mathcal{X} \) and \( Y \in \mathcal{Y} \), the maximum \( \mathcal{E}_1 \) of the overestimation factor when \( k = 1 \) is reached for two Dirac distributions \( X^\delta \) and \( Y^\delta \) respectively centered in \( i \) and \( j \) with \( i \neq j \), which leads to \( \mathcal{E}_1 = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} \chi_i \chi_j = \frac{m_X m_Y}{N} \).
which concludes the proof.

Lemma 8 Let \( X \in \mathcal{X} \) and \( Y \in \mathcal{Y} \) be any two fingerprint vectors. Then the maximum value of the overestimation factor of the sketch codeviation when \( k > 1 \) is exactly

\[
\mathcal{E}_k = \max_{X \in \mathcal{X}, Y \in \mathcal{Y}} \mathcal{E}_k(X, Y) = \frac{m_X m_Y}{N^2} \left( \frac{1}{k} - \frac{1}{N} \right).
\]

Proof: Given \( X \in \mathcal{X} \) and \( Y \in \mathcal{Y} \) any two fingerprint vectors, let us denote \( \mathcal{E}_k^p(X, Y) = \frac{1}{N} \sum_{a \in \rho} \sum_{i, j \in a \setminus \{i\}} x_i y_j + \sum_{i \in a} \sum_{j \in a \setminus \{i\}} x_i y_j \). Consider the partition \( \bar{\rho} = \arg \min_{\rho \in \mathcal{P}_k(\Omega)} \mathcal{E}_k^p(X, Y) \) with \( k > 1 \). We introduce the operator \( \bar{\cdot} \) that operates on fingerprint vectors. This operator is defined as follows

- If it exists \( a \in \bar{\rho} \) such that \( \exists \ell, \ell' \in a \) with \( y_\ell \geq y_{\ell'} \) and \( x_{\ell'} > 0 \), then operator \( \bar{\cdot} \) is applied on the pair \((\ell, \ell')\) of \( X \) so that we have \( x_{\ell'} = x_{\ell'} + 1 \).
- Otherwise, \( \exists a, a' \in \bar{\rho} \) with \( \exists \ell \in a, \exists \ell' \in a', x_\ell > x_{\ell'} > 0 \). Then operator \( \bar{\cdot} \) is applied on the pair \((\ell, \ell')\) of \( X \) so that we have \( x_{\ell'} = x_{\ell'} + 1 \).
- Finally, \( X \) is kept unmodified for all the other items, i.e., \( \forall i \in \Omega \setminus \{\ell, \ell', \ell''\}, x_{\ell''} = x_{\ell''} \).

It is clear that any fingerprint vectors can be constructed from the uniform one, using several iterations of this operator. Thus we split the proof into two parts. The first one supposes that both fingerprint vectors \( X \) and \( Y \) are uniform while the second part considers any two fingerprint vectors.

Case 1. Let \( X_U \) and \( Y_U \) be two uniform fingerprint vectors, i.e., \( X_U = (x_1, \ldots, x_N) \) with \( x_i = \frac{||X_U||_1}{N} \) for 1 \( \leq i \leq N \) and \( Y_U = (y_1, \ldots, y_N) \) with \( y_i = \frac{||Y_U||_1}{N} \) for 1 \( \leq i \leq N \).

We split the analysis into two sub-cases: the class of partitions in which \( x_\ell \) and \( x_{\ell'} \) belong to the same cell \( a \) of a given \( \rho \)-partition \( \rho \), and the class of partitions in which they are located into two separated cells \( a \) and \( a' \). Suppose first that the \( \bar{\cdot} \) operator is applied on \( X_U \). Then the overestimation factor is given by

\[
\mathcal{E}_k(\bar{X}_U, Y_U) = \min(E, E')
\]

with

\[
\begin{aligned}
E &= \min_{\rho \in \mathcal{P}_k(\Omega) \text{ s.t. } \exists a \in \rho, \ell, \ell' \in a} \mathcal{E}_k^p(\bar{X}_U, Y_U) \\
E' &= \min_{\rho \in \mathcal{P}_k(\Omega) \text{ s.t. } \exists a, a' \in \rho, a \neq a' \land \ell \in a \land \ell' \in a'} \mathcal{E}_k^p(\bar{X}_U, Y_U).
\end{aligned}
\]

Let us consider the first term \( E \). We have

\[
E = \min_{\rho \in \mathcal{P}_k(\Omega) \text{ s.t. } \exists a \in \rho, \ell, \ell' \in a} \left( \sum_{b \in a \setminus \{a\}} \sum_{i \in b} \sum_{j \in b \setminus \{i\}} x_i y_j \right)
\]

According to the second term \( E' \), we have

\[
E' = \min_{\rho \in \mathcal{P}_k(\Omega) \text{ s.t. } \exists a, a' \in \rho, a \neq a' \land \ell \in a \land \ell' \in a'} \left( \sum_{b \in a \setminus \{a\}} \sum_{i \in b} \sum_{j \in b \setminus \{i\}} \frac{m_X m_Y}{N^2} \right)
\]

Thus, \( \mathcal{E}_k(\bar{X}_U, Y_U) \leq \mathcal{E}_k(X_U, Y_U) \). By symmetry, we have \( \mathcal{E}_k(X_U, Y_U) \leq \mathcal{E}_k(\bar{X}_U, Y_U) \).

Case 2. In the rest of the proof, we show that for any \( X \) and \( Y \), we have \( \mathcal{E}_k(\bar{X}, Y) \leq \mathcal{E}_k(X, Y) \). Again, we split the proof into two sub-cases according to Relation 4. We get for the first term

\[
\min_{\rho \in \mathcal{P}_k(\Omega) \text{ s.t. } \exists a \in \rho, \ell, \ell' \in a} \mathcal{E}_k^p(\bar{X}, Y)
\]

\[
= \min_{\rho \in \mathcal{P}_k(\Omega) \text{ s.t. } \exists a \in \rho, \ell, \ell' \in a} \left( \mathcal{E}_k^p(X, Y) + \sum_{j \in a \setminus \{\ell, \ell'\}} y_j \right)
\]

\[
= \min_{\rho \in \mathcal{P}_k(\Omega) \text{ s.t. } \exists a \in \rho, \ell, \ell' \in a} \left( \mathcal{E}_k^p(X, Y) + y_{\ell'} - y_{\ell} \right).
\]
For the second term, we have
\[
\min_{\rho \in \mathcal{P}_{\kappa}(\Omega) \cap \mathcal{A}_{\kappa}} \mathcal{E}_k^\rho(\tilde{X}, Y)
\]
\[
= \min_{\rho \in \mathcal{P}_{\kappa}(\Omega) \cap \mathcal{A}_{\kappa}} \left( \mathcal{E}_k^\rho(X, Y) + \sum_{j \in a \setminus \{\ell\}} y_j - \sum_{j \in a' \setminus \{\ell'\}} y_j \right).
\]

By definition of the operator, if it exists $a \in \mathcal{P}$ such that
\[\exists \ell, \ell' \in a, \text{ then } y_\ell \geq y_{\ell'} \text{ and so } E_k^\rho(X, Y) \leq E_k(X, Y).\]
Otherwise, $\ell$ and $\ell'$ are in two separated cells of $\mathcal{P}$, implying that $x_\ell \geq x_{\ell'}$. We then have $\sum_{j \in a \setminus \{\ell\}} y_j \leq \sum_{j \in a' \setminus \{\ell\}} y_j$.

Indeed, suppose by contradiction
\[
x_\ell \sum_{j \in a \setminus \{\ell\}} y_j + x_{\ell'} \sum_{j \in a' \setminus \{\ell\}} y_j < x_\ell \sum_{j \in a \setminus \{\ell\}} y_j + x_{\ell'} \sum_{j \in a' \setminus \{\ell\}} y_j.
\]

Let $\rho'$ be the partition corresponding to the partition $\mathcal{P}$ in which $\ell$ and $\ell'$ have been swapped. Then we obtain $E_k(X, Y) < E_k^\rho(X, Y)$, which is impossible by assumption on $\mathcal{P}$. Thus, in both cases we have $E_k(X, Y) \leq E_k^\rho(X, Y) \leq E_k(X, Y)$. By symmetry, we also have $E_k(X, Y) \leq E_k(X, Y)$.

Thus we have shown that the maximum of any overestimation factor is reached for the uniform fingerprint vector. Lemma 7 concludes the proof.

Case 2 $< k < N$. Let $\bar{\rho} = \arg\min_{\rho \in \mathcal{P}_k(\Omega)} E_k^\rho(X, Y)$, i.e., partition $\bar{\rho}$ minimizes the overestimation factor for a given $k$. Then, there exists a partition $\rho' \in \mathcal{P}_{k+1}(\Omega)$ that can be obtained by splitting a cell of $\mathcal{P}$ in two cells, and constructed as follows
\[
\begin{cases}
\exists a_0 \in \mathcal{P}, \exists a_1, a_2 \in \rho', \text{ such that } a_0 = a_1 \cup a_2 \\
\forall a \in \mathcal{P}, a \neq a_0 \Rightarrow \exists a' \in \rho', \text{ such that } a = a'.
\end{cases}
\]

By using an argument similar to the previous one, we have
\[
E_k(X, Y) \leq E_{k+1}(X, Y) + \frac{1}{N} \left( \sum_{i \in a_1} \sum_{j \in a_2} x_iy_j + \sum_{i \in a_2} \sum_{j \in a_1} x_iy_j \right) \geq E_{k+1}(X, Y).
\]

Lemma 2 concludes the proof.

IV. APPROXIMATION ALGORITHM

In this section, we propose a one-pass algorithm that computes the sketch codeweight between any two large input streams. By definition of the metric (cf. Definition 1), we need to generate all the possible $k$-cell partitions. The number of these partitions follows the Stirling numbers of the second kind, which is equal to $S(N, k) = \frac{1}{k!} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} j^N$. Therefore, $S(N, k)$ grows exponentially with $N$. We show in the following that generating $t = \lceil \log(1/\delta) \rceil$ random $k$-cell partitions, where $\delta$ is the probability of error of our randomized algorithm, is sufficient to guarantee good overall performance of the sketch codeweight metric.

Our algorithm is inspired from the Count-Min Sketch algorithm proposed by Cormode and Muthukrishnan [24]. Specifically, the Count-Min algorithm is an $(\epsilon, \delta)$-approximation algorithm that solves the frequency-estimation problem. For any item $v$ in the input stream $\sigma$, the algorithm outputs an estimation $\hat{x}_v$ of $v$ such that $\mathbb{P}(|\hat{x}_v - x_v| > \epsilon | |X| - x_v| < \delta$, where $\epsilon, \delta > 0$ are given as parameters of the algorithm. The estimation is computed by constructing a two-dimensional array $C$ of $t \times k$ counters through a collection of 2-universal hash functions $\{h_k\}_{1 \leq \ell \leq t}$, where $k = \epsilon / \delta$ and $t = \lceil \log(1/\delta) \rceil$. Each time an item $v$ is read from the input stream, this causes one counter per line to be incremented, i.e., $C[\ell]h[\ell](v)$ is incremented for all $\ell \in [t]$.

To compute the sketch codeweight of any two streams $\sigma_1$ and $\sigma_2$, two sketches $\hat{\sigma}_1$ and $\hat{\sigma}_2$ of these streams are constructed according to the above description (i.e., construction of two arrays $C_{\sigma_1}$ and $C_{\sigma_2}$ of $t \times k$ counters through $t$ 2-universal hash functions $\{h_k\}_{1 \leq \ell \leq t}$). Note that there is no particular assumption on the length of both streams $\sigma_1$ and $\sigma_2$ (their respective length $m_1$ and $m_2$ are finite but unknown). By properties of the 2-universal hash functions $\{h_k\}_{1 \leq \ell \leq t}$, each line $\ell$ of $C_{\sigma_1}$ and $C_{\sigma_2}$ corresponds to the same partition $\rho_\ell$ of $\Omega$, and each entry $a$ of line $\ell$ corresponds to $X_\rho_\ell(a)$ (cf. Definition 1). Therefore, when a query is issued to compute the sketch codeweight $\text{cod}$ between these two streams, the
Algorithm 1: sketch codeviation algorithm

Input: Two input streams $\sigma_1$ and $\sigma_2$; $\delta$ and $\varepsilon$ precision settings;

Output: The sketch codeviation $\hat{\text{cod}}(\sigma_1, \sigma_2)$ between $\sigma_1$ and $\sigma_2$

1. $t \leftarrow \lceil \ln \frac{1}{\delta} \rceil$; $k \leftarrow \lceil \frac{\varepsilon}{\varepsilon} \rceil$;
2. Choose $t$ functions $h : \Omega \rightarrow [k]$, each from a 2-universal hash function family;
3. $C_{\sigma_1}[1..t][1..k] \leftarrow 0$;
4. $C_{\sigma_2}[1..t][1..k] \leftarrow 0$;
5. for $i \in \sigma_1$ do
   6.     for $\ell = 1$ to $t$ do
   7.         $C_{\sigma_1}[\ell][h_\ell(i)] \leftarrow C_{\sigma_1}[\ell][h_\ell(i)] + 1$;
8.     for $j \in \sigma_2$ do
   9.         for $\ell = 1$ to $t$ do
 10.           $C_{\sigma_2}[\ell][h_\ell(j)] \leftarrow C_{\sigma_2}[\ell][h_\ell(j)] + 1$;
11. On query $\text{cod}(\sigma_1, \sigma_2)$ return
       $\min_{1 \leq \ell \leq t} \text{cod}(C_{\sigma_1}[\ell][-], C_{\sigma_2}[\ell][-])$

Thus,
\[
\hat{\text{cod}}[\ell](X, Y) = \frac{1}{N} \sum_{a=1}^{k} \left( \sum_{i=1}^{N} x_i \right) \left( \sum_{i=1}^{N} y_i \right) - \left( \frac{1}{N} \sum_{a=1}^{k} C_{\sigma_1}[\ell][a] \right) \left( \frac{1}{N} \sum_{a=1}^{k} C_{\sigma_2}[\ell][a] \right)
\]

codeviation value between the $\ell^{th}$ line of $C_{\sigma_1}$ and $C_{\sigma_2}$ for each $\ell = 1 \ldots t$ is computed, and the minimum value among these $t$ ones is returned. Figure 1 presents the pseudo-code of our algorithm.

**Theorem 10** The sketch codeviation $\hat{\text{cod}}(X, Y)$ returned by Algorithm 1 satisfies, with $E_{\text{cod}} = \text{cod}(X, Y) - \text{cod}(X, Y)$,

\[
E_{\text{cod}} \geq 0 \quad \text{and} \quad P\left\{ |E_{\text{cod}}| \geq \frac{\varepsilon}{N} (\|X\|_1\|Y\|_1 - \|XY\|_1) \right\} \leq \delta.
\]

**Proof:** The first relation holds by Proposition 3. Regarding the second one, let us first consider the $\ell$-th line of both $C_{\sigma_1}$ and $C_{\sigma_2}$. We have

\[
\hat{\text{cod}}[\ell](X, Y) = \text{cod}(C_{\sigma_1}[\ell][-], C_{\sigma_2}[\ell][-])
\]

By construction of Algorithm 1, $\forall 1 \leq \ell \leq t, \forall i, j \in \sigma_1$ such that $h_\ell(i) = h_\ell(j) = a$, we have

\[
C_{\sigma_1}[\ell][a] = x_i + \sum_{j \neq i} x_j.
\]

Similarly, $\forall 1 \leq \ell \leq t, \forall i, j \in \sigma_2$ such that $h_\ell(i) = h_\ell(j) = a$, we have

\[
C_{\sigma_2}[\ell][a] = y_i + \sum_{j \neq i} y_j.
\]

We have
\[
\mathbb{E} \left[ \hat{\text{cod}}[\ell](X, Y) \right] = \mathbb{E} [\text{cod}(X, Y)] + \frac{1}{N} \sum_{i \neq j} x_i y_j P\{h_\ell(i) = h_\ell(j)\}
\]

By linearity of the expectation, we get
\[
\mathbb{E} \left[ \hat{\text{cod}}[\ell](X, Y) - \text{cod}(X, Y) \right] = \frac{1}{N} \sum_{i \neq j} x_i y_j P\{h_\ell(i) = h_\ell(j)\}
\]

By definition of 2-universal hash functions, we have $P\{h_\ell(i) = h_\ell(j)\} \leq \frac{1}{k}$. Therefore,
\[
\mathbb{E} \left[ \hat{\text{cod}}[\ell](X, Y) - \text{cod}(X, Y) \right] \leq \frac{1}{Nk} \sum_{i \neq j} x_i y_j
\]

By construction of Algorithm 1, we have
\[
\mathbb{E} \left[ \hat{\text{cod}}[\ell](X, Y) \right] \leq \frac{\varepsilon}{eN} (\|X\|_1\|Y\|_1 - \|XY\|_1)
\]

Using the Markov inequality, we obtain
\[
P\left\{ |\hat{\text{cod}}[\ell](X, Y) - \text{cod}(X, Y)| \geq \frac{\varepsilon}{eN} (\|X\|_1\|Y\|_1 - \|XY\|_1) \right\} \leq \frac{1}{e}
\]

By construction $\hat{\text{cod}}(X, Y) = \min_{1 \leq \ell \leq t} \hat{\text{cod}}[\ell](X, Y)$. Thus, by definition of $t$ (cf. Algorithm 1) we obtain
\[
P\left\{ |\hat{\text{cod}}(X, Y) - \text{cod}(X, Y)| \geq \frac{\varepsilon}{e} (\|X\|_1\|Y\|_1 - \|XY\|_1) \right\} \leq \left( \frac{1}{e} \right)^t = \delta
\]
that concludes the proof.

**Lemma 11** Algorithm 1 uses \( O\left(\left(\frac{1}{2}\right)\log \frac{1}{\delta}(\log N + \log m)\right) \) bits of space to give an approximation of the sketch codeviation, where \( m = \max(\|X_i\|_1, \|Y_i\|_1) \).

**Proof:** Both matrices \( C_{\sigma_i} \) for \( i \in \{1, 2\} \) are composed of \( t \times k \) counters, where each counter uses \( O(\log m) \) bits of space. With a suitable choice of hash family, we can store each of the \( t \) hash functions above in \( O(\log N) \) space. This gives an overall space bound of \( O\left(t \log N + tk \log \log m\right) \), which proves the lemma with the chosen values of \( k \) and \( t \).

V. DISTRIBUTED CODEVIAION APPROXIMATION ALGORITHM

In this section, we propose an algorithm that computes the codeviation between a set of \( n \) distributed data streams, so that the number of bits communicated between the \( n \) sites and the coordinator is minimized. This amounts for the coordinator to compute an approximation of the codeviation matrix \( \Sigma \), which is the dispersion matrix of the \( n \) data streams. Specifically, let \( X = \{X_1, X_2, \ldots, X_n\} \) be the set of fingerprint vectors \( X_1, \ldots, X_n \) describing respectively the streams \( \sigma_1, \ldots, \sigma_n \). We have

\[
\tilde{\Sigma} = \left[\tilde{\text{cod}}(X_i, X_j)\right]_{1 \leq i \leq n, 1 \leq j \leq n}.
\]

The algorithm proceeds in rounds until all the data streams have been read in their entirety. In the following, we denote by \( \sigma_i^{(r)} \) the substrum of \( \sigma_i \) received by \( S_i \) during the round \( r \), and by \( d_i \) the number of data items in this substrum.

In a bootstrap phase corresponding to round \( r = 1 \) of the algorithm, each site \( S_i \) computes a single sketch \( C_{\sigma_i} \) of the received data stream \( \sigma_i \) as described in lines 5–7 of Algorithm 1. Once node \( S_i \) has received \( d_i \) data items (where \( d_i \) should typically be set to \( O(100) \)), then node \( S_i \) sends \( C_{\sigma_i} \) to the coordinator, keeps a copy of \( C_{\sigma_i} \), and starts a new round \( r = 2 \). Upon receipt of \( C_{\sigma_i} \) from any \( S_i \), the coordinator asks all the \( n - 1 \) other nodes \( S_j \) to send their own sketch \( C_{\sigma_j} \).

Once the coordinator has received all \( C_{\sigma_i} \), for \( 1 \leq i \leq n \), it sets \( \forall i \in [n], C_{\sigma_i} \leftarrow C_{\sigma_i}^{(1)} \). The coordinator builds the sketch codeviation matrix \( \tilde{\Sigma} = \left[\tilde{\text{cod}}(X_i, X_j)\right]_{1 \leq i \leq n, 1 \leq j \leq n} \) such that the element in position \( i, j \) is the sketch codeviation between streams \( \sigma_i \) and \( \sigma_j \). As the codeviation is symmetric, the codeviation matrix is a symmetric matrix, and thus only the upper-triangle and the diagonal need to be computed.

At round \( r > 1 \), each node \( S_i \) computes a new sketch \( C_{\sigma_i}^{(r)} \) with the sequence of data streams received since the beginning of round \( r \). Let \( d_r = 2d_{r-1} \) be an upper bound on the number of received items during round \( r \). When node \( S_i \) has received at least \( d_{r-1} \) data items, it starts to compute the sketch codeviation between \( C_{\sigma_i}^{(r-1)} \) and \( C_{\sigma_i}^{(r)} \) as in line 11 of Algorithm 1. Once node \( S_i \) has received \( d_r \) data items since the beginning of round \( r \), then it sends its current sketch \( C_{\sigma_i}^{(r)} \) to the coordinator and starts a new round \( r+1 \). Note that during round \( r \), \( S_i \) regularly computes \( \text{cod}(\sigma_i^{(r-1)}, \sigma_i^{(r)}) \) to detect whether significant variations in the stream have occurred before having received \( d_r \) items. This allows to inform the coordinator as quickly as possible that some attack might be undermining. \( S_i \) might then send its current sketch \( C_{\sigma_i}^{(r)} \) to the coordinator once \( \text{cod}(\sigma_i^{(r-1)}, \sigma_i^{(r)}) \) has reached a sufficiently small value. An interesting question left for future work is the study of such a value. Upon receipt of the first \( C_{\sigma_i}^{(r)} \) from any \( S_i \), the coordinator asks all the \( n - 1 \) other nodes \( S_j \) to send their own sketch \( C_{\sigma_j}^{(r)} \). The coordinator locally updates the \( n \) sketches such as \( C_{\sigma_i} \leftarrow C_{\sigma_i} + C_{\sigma_i}^{(r)} \) and updates the codeviation matrix \( \Sigma \) on every couple of sketches.

**Theorem 12** The approximated codeviation matrix \( \Sigma \) returned by the distributed sketch codeviation algorithm satisfies \( \Sigma \geq \Sigma \) and

\[
\mathbb{P}\left\{\left|\Sigma - \tilde{\Sigma}\right| \geq \frac{\epsilon}{N} \max_{i,j \in [n]}(\|X_i\|_1 \|X_j\|_1 - \|X_i X_j\|_1)\right\} \leq \delta.
\]

**Proof:** The statement is derived from Theorem 10 and the fact that the expectation of a matrix is defined as the matrix of expected values.

**Lemma 13** The distributed sketch codeviation algorithm gives an approximation of matrix \( \Sigma \), using \( O\left((1/\epsilon) \log(1/\delta) \log N + \log m\right) \) bits of space for each \( n \) nodes, and \( O\left(n \log m \log(1/\delta) + n\right) \) bits of space for the coordinator, where \( m \) is the maximum size among all the streams, i.e., \( m = \max_{i \in [n]} \|X_i\|_1 \).

**Proof:** From the algorithm definition, each node maintains two sketches with space describes in Lemma 11. The coordinator maintains \( n \) matrices of \( t \times k \) counters and the \( n \times n \) codeviation matrix which takes \( O(n^2 \log m) \) bits, where \( m = \max_{i \in [n]} \|X_i\|_1 \). One can note that the coordinator does not need to maintain the \( t \) hash functions.

**Lemma 14** The distributed sketch codeviation algorithm gives an approximation of matrix \( \Sigma \) by sending \( O\left(\frac{n(1 + (1/\epsilon)) \log(m/2) \log(1/\delta)}{\epsilon}\right) \) bits, where \( r \) is the number of the last round and \( m \) is the maximum size of the streams.

**Proof:** Suppose that the number of rounds of the algorithm is equal to \( r \). At each round, the size of the substrum on each node is at most doubled, and then lower or equal to \( \|X_i\|_2 \). An upper bound of number of bits sent by any node during a round \( r \) is trivially given by \( (1/\epsilon) \log(m/2) \log(1/\delta) \) where \( m = \max_{i \in [n]} \|X_i\|_1 \). Finally, at each end of round, the coordinator sends 1 bit to at most \( n - 1 \) nodes.

VI. PERFORMANCE EVALUATION

We have implemented the distributed sketch codeviation algorithm and have conducted a series of experiments on different types of streams and for different parameters settings. We have fed our algorithm with both real-world data sets and synthetic traces. Real data give a realistic representation of some existing monitoring applications, while the latter ones allow to capture phenomena which may be difficult to obtain from real-world traces, and thus allow to check the robustness...
of our metric. Synthetic traces of streams have been generated from 13 distributions showing very different shapes, that is the Uniform distribution (referred to as distribution 0 in the following), the Zipfian or power law one with parameter \( \alpha \) from 1 to 5 (referred to as distributions 1, \ldots, 5), the Poisson distribution with parameter \( \lambda \) from \( N/2^1 \) to \( N/2^5 \) (distributions 6, \ldots, 11), and the Binomial and the Negative Binomial ones (distributions 12 and 13). All the streams generated from these distributions have a length of around 100,000 items, and contain no more than 1,000 distinct items. Real data have been downloaded from the repository of Internet network traffic [25]. We have used 5 large traces among the available ones. Two of them represent two weeks logs of HTTP requests to the Internet service provider ClarkNet WWW server – ClarkNet is a full Internet access provider for the Metro Baltimore-Washington DC area – the other two ones contain two months of HTTP requests to the NASA Kennedy Space Center WWW server, and the last one represents seven months of HTTP requests to the WWW server of the University of Saskatchewan, Canada. In the following these data sets will be respectively referred to as ClarkNet, NASA, and Saskatchewan traces. We have used as data items the source hosts of the HTTP requests. Table I presents some statistics of these five data traces, in term of stream size (cf. “# items”), number of distinct items in each stream (cf. “# distinct”) and the number of occurrences of the most frequent item (cf. “max. freq.”).

A. Experimental evaluation of the Sketch codeviation

Figures 1 and 2 summarize the results obtained by feeding our distributed codeviation algorithm with respectively syn-

![Figure 1](image1.png)

Figure 1. Synthetic traces – The isopleth on the left has been computed with all the items in memory, while the one on the right has been computed by the distributed algorithm from sketches of length \( k = \log N \).

![Figure 2](image2.png)

Figure 2. Real datasets – The isopleth on the left has been computed with all the items in memory, while the one on the right has been computed by the distributed algorithm from sketches of length \( k = \log N \).

<table>
<thead>
<tr>
<th>Data trace</th>
<th>Trace</th>
<th># items ((m))</th>
<th># distinct ((n))</th>
<th>max. freq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NASA (July)</td>
<td>0</td>
<td>1,891,715</td>
<td>81,983</td>
<td>17,572</td>
</tr>
<tr>
<td>NASA (August)</td>
<td>1</td>
<td>1,569,898</td>
<td>75,058</td>
<td>6,530</td>
</tr>
<tr>
<td>ClarkNet (August)</td>
<td>2</td>
<td>1,654,929</td>
<td>90,516</td>
<td>6,075</td>
</tr>
<tr>
<td>ClarkNet (September)</td>
<td>3</td>
<td>1,673,794</td>
<td>94,787</td>
<td>7,239</td>
</tr>
<tr>
<td>Saskatchewan</td>
<td>4</td>
<td>2,408,625</td>
<td>162,523</td>
<td>52,695</td>
</tr>
</tbody>
</table>

Table I

STATISTICS OF THE FIVE REAL DATA TRACES.
protocol, the distance between the codeviance matrix and the mean of the past ones when all the 10 synthetic traces follow different distributions as a function of the rounds of the protocol, with $\delta = 10^{-5}$.

These results clearly show that our distributed algorithm is capable of efficiently and accurately quantifying how observed data streams change together and in which proportion whatever the shape of the input streams. Indeed, by using sketches of size $k = \log N$, one obtains isopeths very similar to the ones computed with all the items stored in memory. Note that the order of magnitude exhibited by the sketch codeviation matrix is due to the overestimation factor and remains proportional to the exact one. Both results from synthetic traces and real datasets lead to the same conclusions. The following experimental results focus on the detection of attacks.

B. Detection of different profiles of attacks

Figure 3 shows how efficiently our approximation distributed algorithm detects different scenario of attacks in real time. Specifically, we compute at each round of the distributed protocol, the distance between the codeviance matrix $\Sigma$ constructed from the streams under investigation and the mean of covariance matrices $E(\Sigma_N)$ computed under normal situations. This distance has been proposed in [6]. Specifically, given two square matrices $M$ and $M'$ of size $n$, consider the distance as follows:

$$\| M - M' \| = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} (M_{i,j} - M'_{i,j})^2}.$$  

We evaluate at each round $r$, the variable $d_r$ defined by

$$d_r = \| \Sigma_r - E(\Sigma_N) \|.$$  

Interestingly, Jin and Yeung [6] propose to detect abnormal behaviors with respect to normal ones as follows. First they analyze normal network-wide behaviors, and estimate at the end of analysis, a point $c$ and a constant $a$ for $d_r$, satisfying $| d_r - c | < a, \forall r \in \mathbb{N}^*$. The constant $a$ is selected as the upper threshold of the i.i.d $| d_r - c |$. Then when investigating the potential presence of DDoS attacks over the network, they consider as abnormal any traffic pattern that shows for any $r$, $| d_r - c | > a$. Because we think that it is not tractable to characterize what is a normal network-wide traffic a priori, we adapt this definition by considering the past behavior of the traffic under investigation. Specifically, at any round $r > 1$, the distance is computed between the current codeviance matrix $\Sigma_r$ and the mean one $E(\Sigma_r)$ corresponding to previous rounds $1, \ldots, r-1, r$. That is $E(\Sigma_r) = ((r-1)E(\Sigma_{r-1}) + \Sigma_r)/r$. As shown in Figure 3(b), this distance provides better results than the ones obtained with the original distance [6], which is depicted in Figure 3(a).

Based on these distances, we have fed our distributed algorithm with different patterns of traffic. Specifically, Figure 3 shows the distance between the codeviance matrix and the mean ones (respectively based on normal ones for Figure 3(a) and on past ones for Figure 3(b)). These distances are depicted, as a function of time, when the codeviance is exactly computed and when it is estimated with our distributed algorithm with different values of $k$. What can be seen is that, albeit there are up to two orders of magnitude between the exact codeviance matrix and the estimated one, the shape of the codeviance variations are for most of them similar, especially in Figure 3(b). Different attack scenarios are simulated. From round 0 to 10, all the 10 synthetic traces follow the same nominal distribution (e.g., a Poisson distribution). Then from round 10 to 20 a targeted attack is launched by flooding a single node (i.e., one among the ten traces follows a Zipfian
distribution with \( \alpha = 4 \). This gives rise to a drastic and abrupt increase of the distance. As can be shown, the estimated covariance exactly follows the exact one, which is a very good result. Then after coming back to a “normal” traffic, half of the traces are replaced by Zipfian ones (from round 30 to 40), representing a flooding attack toward a group of nodes. As for the previous attack, the covariance matrices are highly impacted by this attack. From round 50 to 60, traces follow a Zipfian distribution with \( \alpha = 1 \) which represents unbalanced network traffic but should not be completely representative of attacks. On the other hand, in the fourth and fifth attack periods, all the traces follow a Zipfian distribution with different values of \( \alpha \geq 2 \), which clearly shows a flooding attack toward a group of targeted nodes.

From these experiments, one could extract the value of the upper threshold \( \alpha \). For instance, \( \alpha \) should be set to 1,000 for the exact codeviation and for the sketch codeviation with \( k = 50 \), which lead to detect all the DDoS attacks. Considering the sketch codeviation with \( k = 10 \) (respectively \( k = 5 \)), \( \alpha \) should be set to 10,000 (respectively 50,000) in order to detect all these attacks.

The main lesson drawn from these results is the good performance of our distributed algorithm whatever the pattern of the attack.

VII. CONCLUSION AND FUTURE WORKS

In this paper we have proposed a novel metric, named the sketch codeviation, that allows to approximate the deviation between any number of distributed streams. We have given upper and lower bounds on the quality of this metric, and have provided an algorithm that additively approximates it using very little space. Beyond its theoretical interest, the sketch codeviation can be exploited in many applications. As discussed in the introducing, large scale monitoring applications are quite straightforward application domains, but we might also use it in publish-subscribe applications, where it must be interesting to track the temporal and spatial correlations that may exist between the different attributes of such applications. This study is planned for future work.

REFERENCES


